



NUMBER: AB17002
July 2017

MDS Orbit Series

GE MDS, LLC. 175 Science Parkway, Rochester, NY 14620 USA
Phone +1 (585) 242-9600, FAX +1 (585) 242-9620 Web: www.gemds.com

Orbit Wireless Bandwidth Optimization

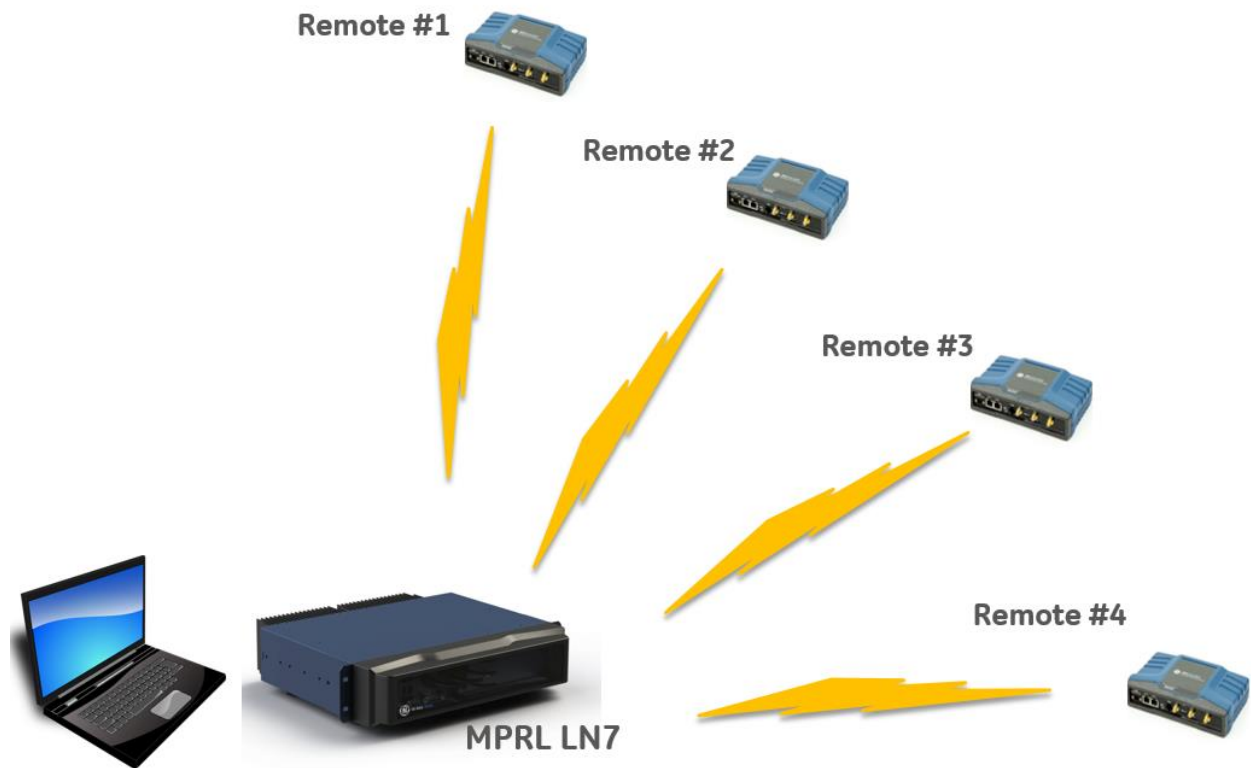
Troubleshooting Tools & Features



Introduction

This document was created to assist with wireless link bandwidth optimization. This will cover using all the features on the Orbit Platform applicable to determining the best configuration to optimize the customer application.

Please note: This document assumes basic knowledge of the Orbit Platform. GE suggests reviewing the YouTube training videos from the link below:
<https://www.youtube.com/watch?v=OcWSG4xERcY&list=PLrbxqFUR561iSD9i6MHBtA6Z692sYr-rq>



Scope

This bulletin is intended for system engineers and end users who are familiar with the Orbit command line interface (CLI) and interested using the Advanced features of the Orbit to accurately control traffic flow to optimize bandwidth.

Firmware Compatibility

This is bulletin is applicable to Orbit MCR devices running firmware version 6.1.2 or greater.

Terms

CLI Command Line Interface
VPN Virtual private Network
DMVPN Dynamic Multipoint VPN
GRE Generic Routing Encapsulation
NHRP Next Hop Resolution Protocol
IPSEC Internet Protocol Security

What is TCPDUMP?

tcpdump is a common packet analyzer that runs under the command line. It allows the user to display TCP/IP and other packets being transmitted or received over a network to which the computer is attached.

This section will cover using the “TCPDUMP” feature to identify traffic to be review and if needed isolated and removed to increase the available bandwidth.

Downloading Putty

This feature can only be run from the command line interface (CLI), so you will need to use a client application like “putty” for example. See link below to download the putty client.

<http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html>

Logging into the CLI

1. Once you download and open the putty client it will look like the image below.

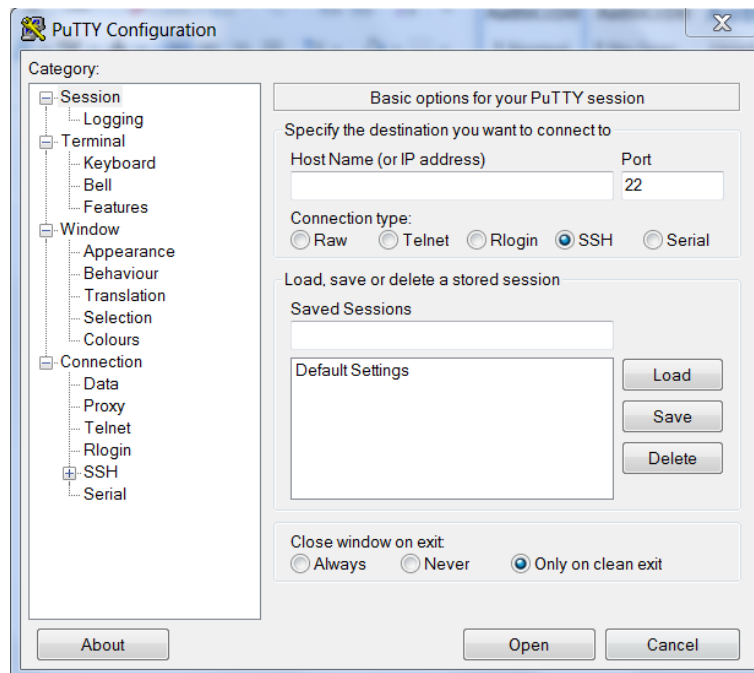


Image-2.21

2. You will fill out the connection variables based on the type of connection. There are two options, serial and IP. For Serial, you will be using the COM port which will be defined by the device manager and the default baud-rate on the orbit USB port is 115200 and using a serial cable will have a default baud-rate 9600.

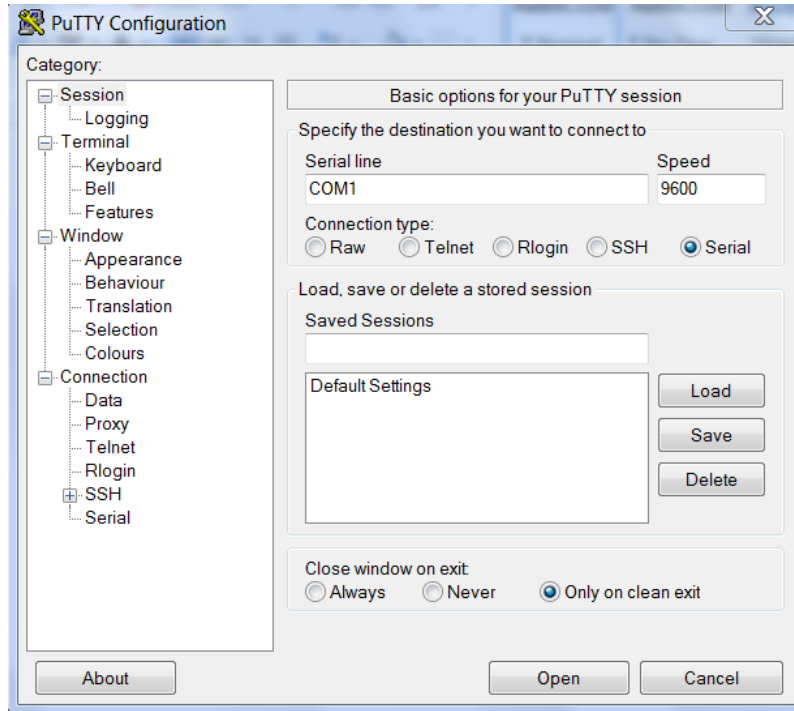


Image-2.22

- To connect via IP address, the Orbit Platform default IP address is 192.168.1.1. If the IP address is something different then that is the address you should use and the default port used for SSH is port 22.

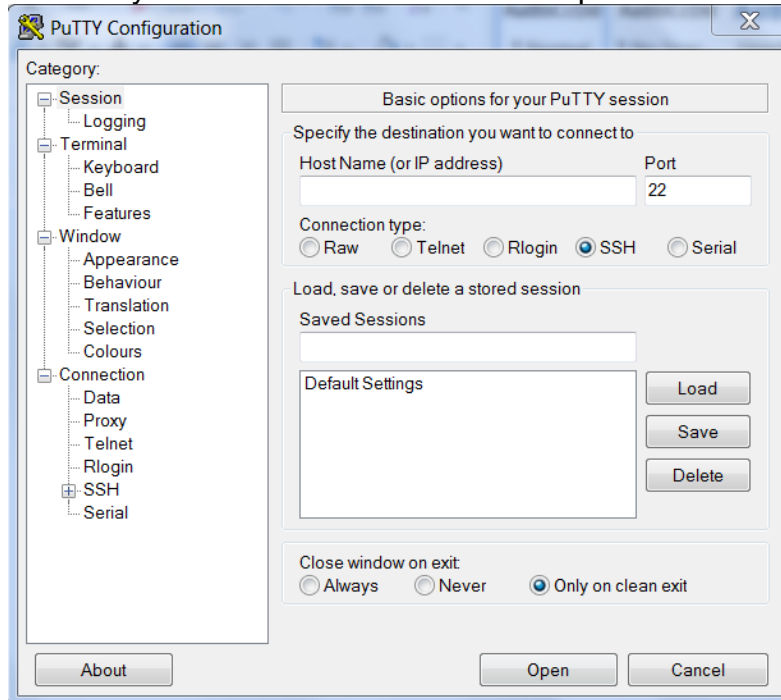


Image-2.23

- The default username and password for the Orbit Platform is “admin”.

```
login as: admin
admin@192.168.1.1's password:
```

- To prevent the CLI from automatically logging out due to inactivity GE recommends using the command “ set idle-timeout 0”.

```
admin connected from 192.168.1.8 using ssh on (none)
admin@(none) 22:56:10> set idle-timeout 0
[ok] [2017-03-21 22:56:35]
admin@(none) 22:56:35> █
```

Using TCPDUMP on the Orbit

1. Once logged onto the CLI, the Wireshark feature is not accessible by default. You will need to enter “unhide debug” to make this command available from the operator prompt.

```
admin@(none) 22:56:35> unhide debug
[ok] [2017-03-21 23:03:48]
admin@(none) 23:03:48> █
```

2. After the feature is accessible, enter the command “tcpdump list”. This will show all the interfaces available on the Orbit unit you are logged into.

```
[ok] [2017-03-21 23:03:48]
admin@(none) 23:03:48> tcpdump list
1.fec0 [Up, Running]
2.br0 [Up, Running]
3.wlan0 [Up, Running]
4.mon.wlan0 [Up, Running]
5.eth5 [Up, Running]
6.any (Pseudo-device that captures on all interfaces) [Up, Running]
7.lo [Up, Running, Loopback]
8.eth0 [Up]
9.eth1 [Up]
10.eth3 [Up]
11.eth4 [Up]
12.nflog (Linux netfilter log (NFLOG) interface)
13.nfqueue (Linux netfilter queue (NFQUEUE) interface)
[ok] [2017-03-21 23:06:08]
admin@(none) 23:06:08> █
```

3. Using the screenshot above as an example these interfaces are:
admin@(none) 23:03:48> tcpdump list
1.fec0 [Up, Running] # This interface does not monitor usable information
2.br0 [Up, Running] # This represents the bridge interface which handles L2 switching
3.wlan0 [Up, Running] # This represents the wireless interface
4.mon.wlan0 [Up, Running]
5.eth5 [Up, Running] # This represents the cellular interface
6.any (Pseudo-device that captures on all interfaces) [Up, Running] # catch-all that includes all traffic
7.lo [Up, Running, Loopback] # only applicable if a loop[back interface is used in your current configuration
8.eth0 [Up] # Ethernet interfaces #1
9.eth1 [Up] # Ethernet interfaces #2
10.eth3 [Up] # Ethernet interfaces #3
11.eth4 [Up] # Ethernet interfaces #4
12.nflog (Linux netfilter log (NFLOG) interface) # is not used for IP traffic

13.nfqueue (Linux netfilter queue (NFQUEUE) interface) # is not used for IP traffic

- Each exchange of IP data will include the time stamp, IP address and port from the source device and the same information about the destination. An example of multiple data streams is demonstrated in the image below.

```
admin@(none) 23:15:45> tcpdump interface eth5
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth5, link-type EN10MB (Ethernet), capture size 262144 bytes
23:16:30.184777 IP 12.229.99.64.17472 > 166.239.44.20.63498: Flags [.], seq 2341867765:2341869153, ack 3423439306, win 260, length 1388
23:16:30.184780 IP 12.229.99.64.17472 > 166.239.44.20.63498: Flags [.], seq 1388:2776, ack 1, win 260, length 1388
23:16:30.188107 IP 166.239.44.20.63498 > 12.229.99.64.17472: Flags [.], ack 2776, win 45804, length 0
23:16:30.310160 IP 12.229.99.64.17472 > 166.239.44.20.63498: Flags [.], seq 0:1388, ack 1, win 260, length 1388
23:16:30.310517 IP 12.229.99.64.17472 > 166.239.44.20.63498: Flags [.], seq 1388:2776, ack 1, win 260, length 1388
23:16:30.310522 IP 12.229.99.64.17472 > 166.239.44.20.63498: Flags [.], seq 2776:4164, ack 1, win 260, length 1388
23:16:30.315492 IP 166.239.44.20.63498 > 12.229.99.64.17472: Flags [.], ack 2776, win 45804, options [nop,nop,sack 1 {0:1388}], length 0
23:16:30.317587 IP 166.239.44.20.63498 > 12.229.99.64.17472: Flags [.], ack 2776, win 45804, options [nop,nop,sack 1 {1388:2776}], length 0
23:16:30.454512 IP 12.229.99.64.17472 > 166.239.44.20.63498: Flags [.], seq 4164:5552, ack 1, win 260, length 1388
23:16:30.454517 IP 12.229.99.64.17472 > 166.239.44.20.63498: Flags [.], seq 5552:6940, ack 1, win 260, length 1388
23:16:30.456895 IP 166.239.44.20.63498 > 12.229.99.64.17472: Flags [.], ack 6940, win 44763, length 0
23:16:30.717364 IP 12.229.99.64.17472 > 166.239.44.20.63498: Flags [.], seq 6940:8328, ack 1, win 260, length 1388
23:16:30.717729 IP 12.229.99.64.17472 > 166.239.44.20.63498: Flags [.], seq 8328:9716, ack 1, win 260, length 1388
23:16:30.717734 IP 12.229.99.64.17472 > 166.239.44.20.63498: Flags [.], seq 9716:11104, ack 1, win 260, length 1388
23:16:30.717739 IP 12.229.99.64.17472 > 166.239.44.20.63498: Flags [P.], seq 11104:12492, ack 1, win 260, length 1388
23:16:30.723846 IP 166.239.44.20.63498 > 12.229.99.64.17472: Flags [.], ack 12492, win 44416, length 0
23:16:30.968649 IP 12.229.99.64.17472 > 166.239.44.20.63498: Flags [.], seq 12492:13880, ack 1, win 260, length 1388
23:16:30.968654 IP 12.229.99.64.17472 > 166.239.44.20.63498: Flags [.], seq 13880:15268, ack 1, win 260, length 1388
23:16:30.969167 IP 12.229.99.64.17472 > 166.239.44.20.63498: Flags [.], seq 15268:16656, ack 1, win 260, length 1388
23:16:30.969438 IP 12.229.99.64.17472 > 166.239.44.20.63498: Flags [.], seq 16656:18044, ack 1, win 260, length 1388
23:16:30.969442 IP 12.229.99.64.17472 > 166.239.44.20.63498: Flags [.], seq 18044:19432, ack 1, win 260, length 1388
23:16:30.972596 IP 166.239.44.20.63498 > 12.229.99.64.17472: Flags [.], ack 18044, win 44416, length 0
23:16:31.184909 ARP, Request who-has 12.229.99.64 tell 166.239.44.20, length 28
```

- Using the live data being streamed across your screen, highlight a single stream of data like the image below. As shown in the image below there is a timestamp, IP address and port pointed at the destination. The destination of the traffic depends on your perspective.

```
22:20:00.654915 ARP, Request who-has 165.156.25.115 tell 166.239.44.20, length 28
22:20:00.655683 ARP, Reply 165.156.25.115 is-at 02:50:f3:00:00:00, length 28
22:20:01.780512 IP 166.239.44.20.52101 > 104.129.194.41.18000: Flags [.], seq 171849498:171849499, ack 4279000154, win 4164, length 1
22:20:01.846392 IP 104.129.194.41.18000 > 166.239.44.20.52101: Flags [.], ack 1, win 2100, length 0
22:20:02.572492 IP 104.129.194.41.18000 > 166.239.44.20.52098: Flags [.], ack 3896874170, win 2100, length 0
22:20:02.576132 IP 166.239.44.20.52098 > 104.129.194.41.18000: Flags [.], ack 1, win 3948, length 0
[ok] [2017-03-22 22:20:02]
admin@(none) 22:20:02> █
```

- Use the modifier character (“| match”) to modify the traffic per data passing that can be matched for example:
 - tcpdump interface br0 | match 192.168.1.1** (matching a specific IP address)
 - tcpdump interface eth5 | match 192.168.1.1.22** (matching a specific IP address and port)

Orbit Firewall

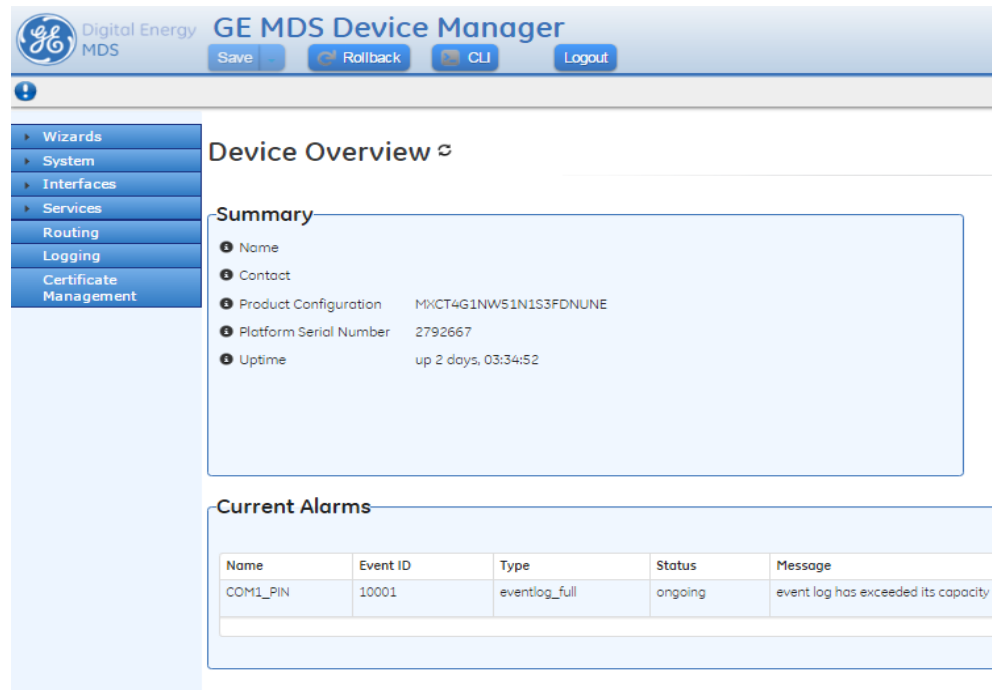
This section will take you through using firewall rules block unwanted traffic from the wireless links. We will go through this from the web interface.

Making Firewall Rules with the Web UI

The recommended method to create firewall rules on the Orbit Platform is to use the access control list wizard which will guide you step by step through creating firewall rules.

1. From the “Device Overview” screen or home screen navigate to the services screen.

DEVICE OVERVIEW > SERVICES > FIREWALL



GE MDS Device Manager

Save Rollback CLI Logout

Wizards
System
Interfaces
Services
Routing
Logging
Certificate Management

Device Overview

Summary

- Name
- Contact
- Product Configuration: MxCT4G1NW51N1S3FDNUNE
- Platform Serial Number: 2792667
- Uptime: up 2 days, 03:34:52

Current Alarms

Name	Event ID	Type	Status	Message
COM1_PIN	10001	eventlog_full	ongoing	event log has exceeded its capacity

IMAGE-3.1.1- DEVICE OVERVIEW

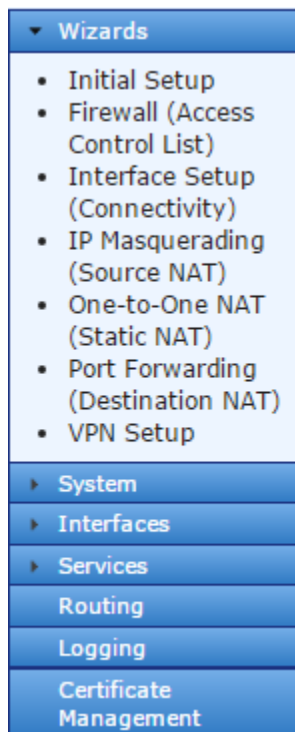


IMAGE 3.1.2-FEATURE LIST

2. Once on the access-list wizard navigate move to the “IN_UNTRUSTED” firewall filter. This is the primary filter you will need to apply to remove unwanted traffic. In most cases the unwanted traffic will likely be UDP broadcast traffic or standard UDP traffic. In most cases TCP is used in polling situations but GE recommends discussion with your network admin or system integrator before blocking traffic.

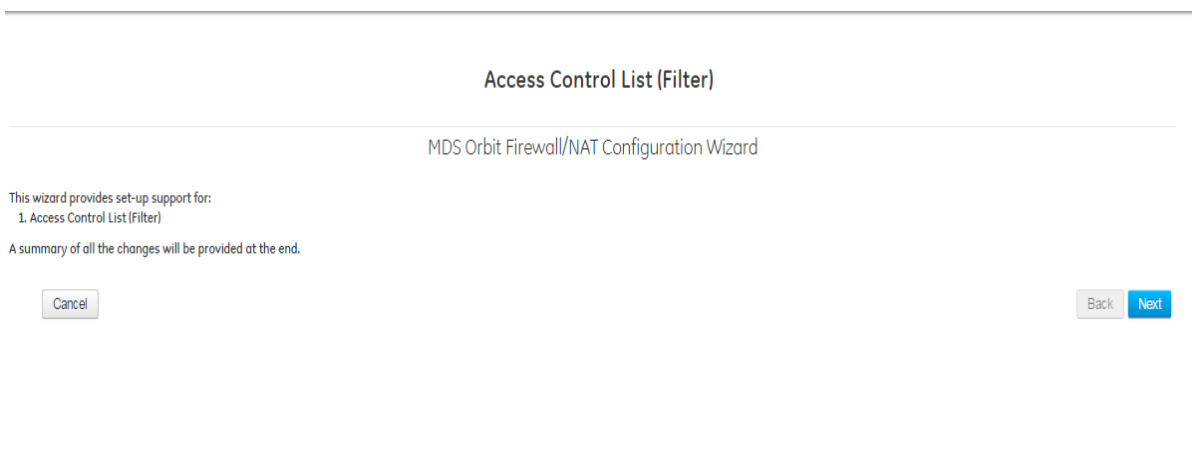


IMAGE-3.1.3- ACCESS CONTROL LIST (FILTER)

- Now, check the box next to the “IN_UNTRUSTED” firewall filter which will be normally used to filter inbound traffic coming into the Orbit. Click next to take you to the filter modification screen.

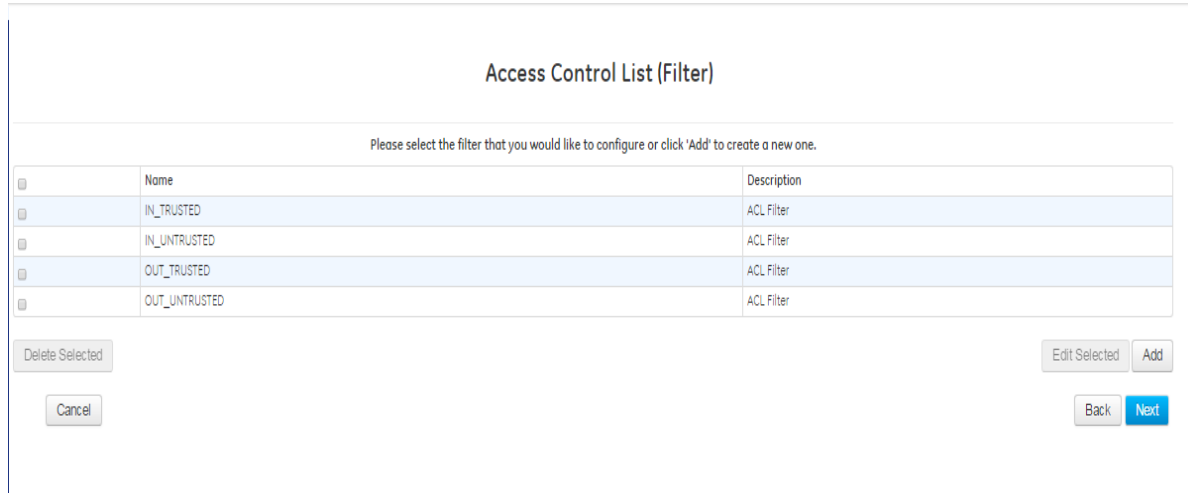
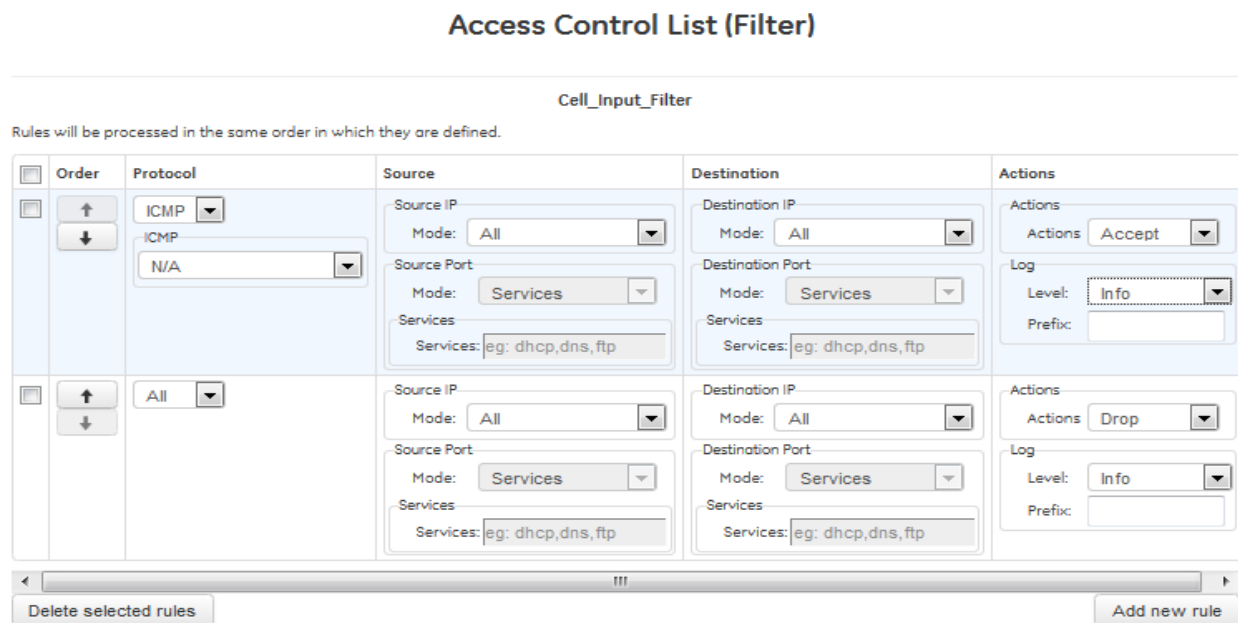


IMAGE-3.1.4- “ACCESS CONTROL LIST (FILTER) IN_UNTRUSTED”

- Once you are on the “IN_UNTRUSTED” modification screen you can create, delete, and or modify firewall rules.



5. The following options are available for Firewall rules.

- **Order** – Click the arrows to sort rules in order of priority. Rules with higher priority are applied before rules with lower priority; rule sets containing more than one rule should be sorted accordingly.
- **Protocol** – *All, SCTP, TCP, UDP, ICMP, ESP*. Specifies the IP protocol of traffic that the rule should be applied to.
- **ICMP** - When selected, the rule will only apply to that specific ICMP message only. For ICMP message type definitions, see *RFC792*, available from the Internet Engineering Task Force, <http://www.ietf.org>
- N/A - the rule will be applied to all ICMP protocol messages.
- Destination Unreachable
- Echo Request
- Echo Reply
- Address Mask Request
- Address Mask Reply
 - - Parameter Problem
 - - Redirect
 - - Router Advertisement
 - - Router Solicitation
 - - Source Quench
 - - Time Exceeded
 - - Timestamp Request
 - - Timestamp Reply.
- **Source IP** – Apply rule to traffic that originates at a specific source address or addresses.
- **Mode** – Address, Address Range, Address Set, Not Address, Not Address Range, Not Address Set.
 - - All – Apply rule regardless of source address.
 - - Address - Apply rule to a specific source address and prefix.
 - - Address Range – Apply rule to a range of source addresses.
 - - Address Set – Apply rule to a non-contiguous set of source addresses.
 - - Not Address - Apply rule to traffic that does *not* originate from a specific source address and prefix.

- - Not Address Range – Apply rule to traffic that does *not* originate from a source address range.
- - Not Address Set – Apply rule to traffic that does *not* originate from a non-contiguous set of source addresses.
- **Destination Port** – Apply rule to traffic intended for a specific destination port. This option is available only with protocols *SCTP, TCP, and UDP*.
- **Services** – Services, Port Range, Not Services, Not Port Range.
- **Services** – Apply rule to traffic intended for one or more designated well-known service destination ports. The services must be specified by name and separated by commas.
- - Port Range – Apply rule to traffic intended for a specific destination port or set of ports.
- - Not Services – Apply rule to traffic that is *not* intended for one or more designated well-known service destination ports. The services must be specified by name and separated by commas.
- - Not Port Range – Apply rule to traffic that is *not* intended for a specific destination port or set of ports.
- **Actions** – *Accept, Drop, Reject*. Specifies what should be done with packets that match the rule.
 - - Accept – Allow packets to ingress or egress the unit.
 - - Drop – Block packets from ingress or egress.
 - - Reject – Block packets from ingress or egress and send an error message to the sender. When *ICMP* protocol is selected, a rejection message may be chosen.
 - - Reject Type – Net unreachable, Host unreachable, Port unreachable, Proto unreachable, Net prohibited, Host prohibited, Admin prohibited
- **Log** – *Optional*. Allows packets that meet the rule to be logged to the event log.
- **Level** – Emergency, Alert, Critical, Error, Warning, Notice, Info, Debug.
- **Prefix** – Enter a text string to prepend to generated log entries.
- **Allow Select Cell Inbound traffic**
- In this example, the input filter will be restrictive and permit only some types of traffic: IPsec tunnel traffic, UDP services DNS, NTP, and IKE (to allow IPsec connection setup), and TCP

services SSH and NETCONF (to allow management of the MCR).

- To create a rule to permit IPsec tunnel traffic, select **Protocol** ESP and ensure that **Action** is set to Accept. The **Log Level** can be set to Debug, unless incoming IPsec traffic is of interest.

Orbit Quality of Service

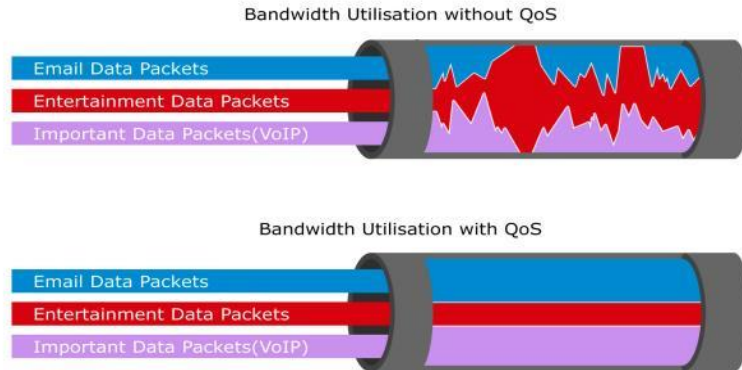
This section will cover quality of service and how it can be applied after variables are identified with “TCPDUMP” and after excess traffic was filtered with the firewall rules.

What is QOS:

- A set of networking capabilities for management of traffic flows in a network
- Orbit supports Egress QoS with up to 16 Queues
- Inspection
- Classification
- Prioritization
- Marking
- Queuing
- Shaping

1. Where Used, Benefits:

- Benefits shows in congested networks that carry multiple applications or flows of traffic
- Quality of Service enables operators to configure the network in such a way that critical traffic transits first in a congested network, with the end goal of reducing its latency and improving throughput
- QoS is especially beneficial in low bandwidth networks such as narrowband



2. QoS ensures that important traffic is sent out first, to reduce its latency and probability of being dropped due to running out of buffers.
 - **Classification:** incoming packets are inspected and classified based on Layer 2,3 or 4 header information (matching specific applications), and then assigned classes. Classifiers may include Ethertype, VLAN ID, 802.1p, source/destination IP, DSCP, TOS, TCP/UDP port numbers etc...
 - **Prioritization:** Each class is then assigned a priority with local significance. Orbit supports 16 priorities, and the lower the number, the higher the priority/importance.
 - **Shaping:** allocation of dedicated (or dynamic) uplink bandwidth% on a per class and application basis. I.e., SCADA = 20%, VoIP = 30%, Video = 35% etc...
 - **Priority Queuing (Strict):** Traffic in a higher priority queue keeps on getting processed/switched until exhausted, then Orbit moves on to process packets from the queue next in line

- **Fair Queuing:** An internal algorithm runs between the various queues to ensure that severe congestion on one specific queue does not prevent other queues from transmitting.



QoS Configuration

STEP 1 / Configuring:

- In the web UI, the QoS service is configured under **QoS Service** → **Basic Config**

QoS Service

Status	Basic Config	Advanced Config	Actions
<p>▼ General</p> <p><input checked="" type="checkbox"/> Enabled <input checked="" type="checkbox"/></p> <p>▶ Policy</p> <p>▶ Classifier</p>			

- To create a classifier for GOOSE messages, click **Add** in the **Classifier** submenu. The **Configure Classifier Details** appears.

Configure Classifier Details

Name*

3. Configuring:

- Give the new classifier a name and click the Add button. A menu bearing the classifier's name appears to configure it.

Classifier

Search x Add ... Delete

Name	Match Type	Metric
Table is empty		

Configure Classifier Details

Match Type: Any

Match

Search x Add ... Delete

Name	Ethernet - Src Address - Address	Ethernet - Dst Address - Address	Ethernet - Vlan Priority - Priority	Ethernet - Vlan ID - ID
Match1				

Showing 1 to 1 of 1

Metric: 10

Finish

4. Configuring:

- Click the Add button under the Match submenu to add a match rule.

Configure Match Details

0 Name*

- First, give the new match rule a name and click the **Add** button.

Configure Match Details

IPv4

Protocol

Src Address

Dst Address

Tos Dscp

Ethernet

Ether Type

Src Address

Dst Address

5. Configuring:

- A match rule can be created to classify on either IPv4 or Ethernet. In this example, we use ether type to classify GOOSE messages.
- To classify based on ether type, click the check box to the left of Ether Type.

Ethernet

Ether Type

Not

Type

Protocol

- To specify GOOSE messages, click the Type dropdown box and select Protocol. Next, select Goose from the Protocol drop-down box. Save the configuration.

- Now create a QoS policy that applies this classifier. Return to the QoS Services ---> Basic Config menu and click the Add button in the Policy submenu.

6. Configuring:



Configure Policy Details

0 Name* Policy1

Add Cancel

- First, give the new policy a name and click the **Add** button



Configure Policy Details

Type

Choices

Finish

- The Type dropdown contains the following options.
 - Prioritization
 - Fairness
 - Shaping-htb

7. Configuring:

- Since this example prioritizes GOOSE messages above all other traffic, select Prioritization.

Configure Policy Details

Type

Choices*

Default Priority* 5

Class

Search x Add ... Delete

Name	Priority	Next Policy
HighPriority	1	

Showing 1 to 1 of 1

Finish

- The **Default** priority will be applied to all packets that do not match any priority class in the policy. The value can be a number from 1-16, where 1 is the highest priority and 16 is the lowest. For this example, we chose 5.

8. Configuring:

- Click the **Add** button in the class submenu to create a new priority class. The **Configure Class Details** appears.

Configure Class Details

Name* HighPriority

Add Cancel

- Enter a name for the priority class and click **Add**. A menu bearing the policy's name appears.

Configure Class Details

Priority*

Classifier

Next Policy

[Finish](#)

9. Configuring:

- Select the name of the classifier that was created for this example from the **Classifier** dropdown box. This incorporates the classifier, which selects all GOOSE messages, into the new priority class. Since this example makes GOOSE messages the highest priority, enter 1 as the priority. Click **Save**. The configured QoS classifiers and policies are listed at **QoS Services ---> Basic Config**.

QoS Service [↻](#)

[Status](#) | [Basic Config](#) | [Advanced Config](#) | [Actions](#)

General
 Enabled

Policy
 Search Add ... Delete [Icons]

Name
Policy1

Showing 1 to 1 of 1

Classifier
 Search Add ... Delete [Icons]

Name	Match Type	Metric
Example	any	10

Showing 1 to 1 of 1

10. Configuring:

- This policy should be applied to an interface before it has any effect. Navigate to **Interfaces** and click on the desired interface. Click on the **QoS** dropdown from the **Basic Config** tab and select the new QoS policy to apply it to all traffic leaving that interface. Once configuration is complete, click **Save**.

ETH1 Interface

Status	Basic Config	Advanced Config	Actions
--------	--------------	-----------------	---------

▸ General
▸ IPv4
▸ Filter
▸ Nat
▾ QoS

QoS

Output

EXAMPLE CLI Configuration

- Below is an example configuration to give management vlan priority over other data vlans

#Remove previous qos rules

```
delete services qos policy
delete services qos classifier
```

#Create qos rules priority 1. VLAN1 2. VLAN20 3. VLAN300.

```
set services qos enabled true
set services qos policy Policy1 prioritization default-priority 15
set services qos policy Policy1 prioritization class HIGH priority 1
set services qos policy Policy1 prioritization class HIGH classifier [
VLAN1 ]
set services qos policy Policy1 prioritization class LOW priority 15
set services qos policy Policy1 prioritization class LOW classifier [
VLAN300 ]
set services qos classifier ICMP match M1 ipv4 protocol
set services qos classifier ICMP match M1 ipv4 protocol assigned-
number icmp
set services qos classifier VLAN1 match M1 ethernet ether-type
set services qos classifier VLAN1 match M1 ethernet ether-type
protocol vlan
set services qos classifier VLAN1 match M1 ethernet vlan-id
set services qos classifier VLAN1 match M1 ethernet vlan-id id 1
set services qos classifier VLAN300 match M1 ethernet ether-type
set services qos classifier VLAN300 match M1 ethernet ether-type
protocol vlan
set services qos classifier VLAN300 match M1 ethernet vlan-id
set services qos classifier VLAN300 match M1 ethernet vlan-id id 300
```

#Apply the QOS rules to the Ln interface

#On MCR/ECR

```
set interfaces interface LnRadio qos output Policy1
```

#On MPRL

```
set interfaces interface Inms qos output Policy1
```

Example#2 configuration

- Below is an example configuration to give data VLAN priority over other VLANs
- Additionally, this configuration uses traffic shaping to individually isolate the amount of bandwidth that will be used for each VLAN defined
- **#Remove previous qos rules**
delete services qos policy
delete services qos classifier
- **#ADD QOS Prioritization Policy**
set services qos enabled true
set services qos policy Policy1 prioritization default-priority 15
set services qos policy Policy1 prioritization class HIGH priority 1
set services qos policy Policy1 prioritization class HIGH classifier [VLAN1]
set services qos policy Policy1 prioritization class LOW priority 15
set services qos policy Policy1 prioritization class LOW classifier [VLAN300]
- **#ADD QOS Traffic shaping Policy2**
set services qos policy Policy2 shaping-htb
set services qos policy Policy2 shaping-htb default-class LOW
set services qos policy Policy2 shaping-htb committed-rate 100
set services qos policy Policy2 shaping-htb class HIGH committed-rate 80
set services qos policy Policy2 shaping-htb class HIGH max-rate 120
set services qos policy Policy2 shaping-htb class HIGH priority 1
set services qos policy Policy2 shaping-htb class HIGH classifier [VLAN300]
set services qos policy Policy2 shaping-htb class LOW committed-rate 10
set services qos policy Policy2 shaping-htb class LOW max-rate 10
set services qos policy Policy2 shaping-htb class LOW priority 15
set services qos policy Policy2 shaping-htb class LOW next-policy Policy1
set services qos policy Policy2 shaping-htb class MEDIUM committed-rate 40
set services qos policy Policy2 shaping-htb class MEDIUM max-rate 40
set services qos policy Policy2 shaping-htb class MEDIUM priority 10
set services qos policy Policy2 shaping-htb class MEDIUM classifier [VLAN1]

```
set services qos policy Policy2 shaping-htb class MEDIUM next-policy
Policy1
```

- **#ADD definition of traffic classifiers**

```
set services qos classifier ICMP match M1 ipv4 protocol
set services qos classifier ICMP match M1 ipv4 protocol assigned-
number icmp
set services qos classifier VLAN1 match M1 ethernet ether-type
set services qos classifier VLAN1 match M1 ethernet ether-type
protocol vlan
set services qos classifier VLAN1 match M1 ethernet vlan-id
set services qos classifier VLAN1 match M1 ethernet vlan-id id 1
set services qos classifier VLAN300 match M1 ethernet ether-type
set services qos classifier VLAN300 match M1 ethernet ether-type
protocol vlan
set services qos classifier VLAN300 match M1 ethernet vlan-id
set services qos classifier VLAN300 match M1 ethernet vlan-id id 300
```


IPERF testing

This section will review how to use Iperf to determine bandwidth.

What is IPERF?

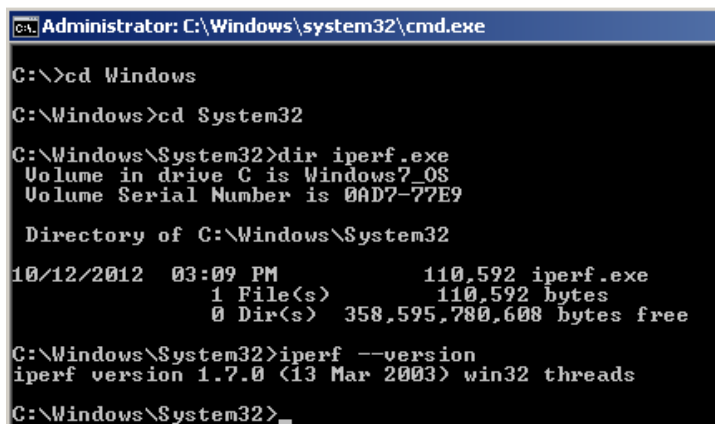
Troubleshooting network related issues can be challenging at times. One great tool for on-the-spot testing of network bandwidth is Iperf. It's a simple, yet powerful tool that can provide visibility on network performance metrics such as bandwidth, latency (delay), jitter, and packet loss. If you are not comfortable with the command prompt, the program is available in a GUI version called Jperf too.

Iperf measures network performance in terms of bandwidth, but its worth noting that the unit of measurement for file size and transfer rate are different. Files are typically stated in bytes, while transfer speeds are expressed in bits

When testing bandwidth performance with Iperf, what we're actually testing is maximum TCP bandwidth at the transport layer (L4). Lots of applications use TCP as the transport protocol, include HTTP, SMTP, FTP, etc. Unlike UDP (another commonly used L4 protocol) TCP is a reliable, connection-oriented protocol with built-in mechanisms for connection established, acknowledgement, and termination. This connection management is why TCP is used (and not UDP) to test bandwidth. If UDP was being used, any lost packet(s) would go missing from the traffic flow and not be resent. Simply put, TCP has a means to detect and retransmit any lost segments. If your network is configured for QoS or CoS, you must make sure that the IPs or ports for these tests are configured for the highest priority QoS. Otherwise, your results will not be accurate.

Bandwidth Verification

The following examples use a combination of Microsoft Windows 7 (64-bit) and XP (32-bit). Also, the default TCP window size is used but keep in mind you can adjust window size and effect bandwidth results. To use the program, open the command prompt and navigate to the directory where Iperf.exe is located (Start>Run and type cmd, then press enter). Once in the command prompt you can change directories via the cd command. This example uses Iperf version 1.7.0.



```
CA Administrator: C:\Windows\system32\cmd.exe
C:\>cd Windows
C:\Windows>cd System32
C:\Windows\System32>dir iperf.exe
Volume in drive C is Windows7_OS
Volume Serial Number is 0AD7-77E9

Directory of C:\Windows\System32

10/12/2012  03:09 PM                110,592 iperf.exe
             1 File(s)                110,592 bytes
             0 Dir(s)  358,595,780,608 bytes free

C:\Windows\System32>iperf --version
iperf version 1.7.0 (13 Mar 2003) win32 threads
C:\Windows\System32>_
```

Before we get started lets quickly discuss Iperf's architecture. Iperf uses a client/server model, where traffic is initiated from the client and traverses the network (LAN and/or WAN) to the server. So, when testing bandwidth in both directions we'll need to run the test twice, once in each direction. Also, by default the server will listen for the client on TCP port 5001 but like most options this can be changed. Note the ability to change TCP ports in Iperf opens avenues to use it for testing open TCP ports. Thus, if a firewall or filtering device is blocking port(s) on a segment of your network you can use Iperf to verify this. To get started we'll setup the server side first, entering `iperf -s -i 1`. The `-s` command designates this workstation as the "server" and the `-i 1` command sets the console output interval to one second.

```

C:\Windows\system32\cmd.exe - iperf -s -i 1

C:\Windows\System32>iperf -s -i 1

-----
Server listening on TCP port 5001
TCP window size: 8.00 KByte (default)
-----
_

```

Now to start a test, go onto the client and enter `iperf -c <server_ip> -i 1`. The `-c <server_ip>` command designates this workstation as the "client" and the IP address of the server. By default, the test runs for 10 secs but can be changed with the `<server_ip>-t` option. In the example below a test was run (from client to server) displaying the transfer rate (MB) and bandwidth (Mb) performance every second.

```

C:\WINDOWS\system32\cmd.exe

C:\WINDOWS\system32>iperf -c 10.0.0.106 -i 1

-----
Client connecting to 10.0.0.106, TCP port 5001
TCP window size: 63.0 KByte (default)
-----
[1912] local 10.0.0.40 port 1105 connected with 10.0.0.106 port 5001
[ ID] Interval      Transfer      Bandwidth
[1912] 0.0- 1.0 sec    11.0 MBytes   92.4 Mbits/sec
[1912] 1.0- 2.0 sec    11.1 MBytes   93.3 Mbits/sec
[1912] 2.0- 3.0 sec    10.5 MBytes   88.4 Mbits/sec
[1912] 3.0- 4.0 sec    11.0 MBytes   92.7 Mbits/sec
[1912] 4.0- 5.0 sec    11.1 MBytes   93.4 Mbits/sec
[1912] 5.0- 6.0 sec    11.1 MBytes   93.2 Mbits/sec
[1912] 6.0- 7.0 sec    11.0 MBytes   92.5 Mbits/sec
[1912] 7.0- 8.0 sec    11.0 MBytes   92.3 Mbits/sec
[1912] 8.0- 9.0 sec    11.1 MBytes   93.1 Mbits/sec
[1912] 9.0-10.0 sec   11.1 MBytes   93.2 Mbits/sec
[1912] 0.0-10.0 sec   110 MBytes    92.3 Mbits/sec
C:\WINDOWS\system32>_

```

Now if we look back at the server you'll also see similar output as the client. The test output shows Iperf could transfer 110MBytes of data at a rate of 92.3Mbits, from client to server.

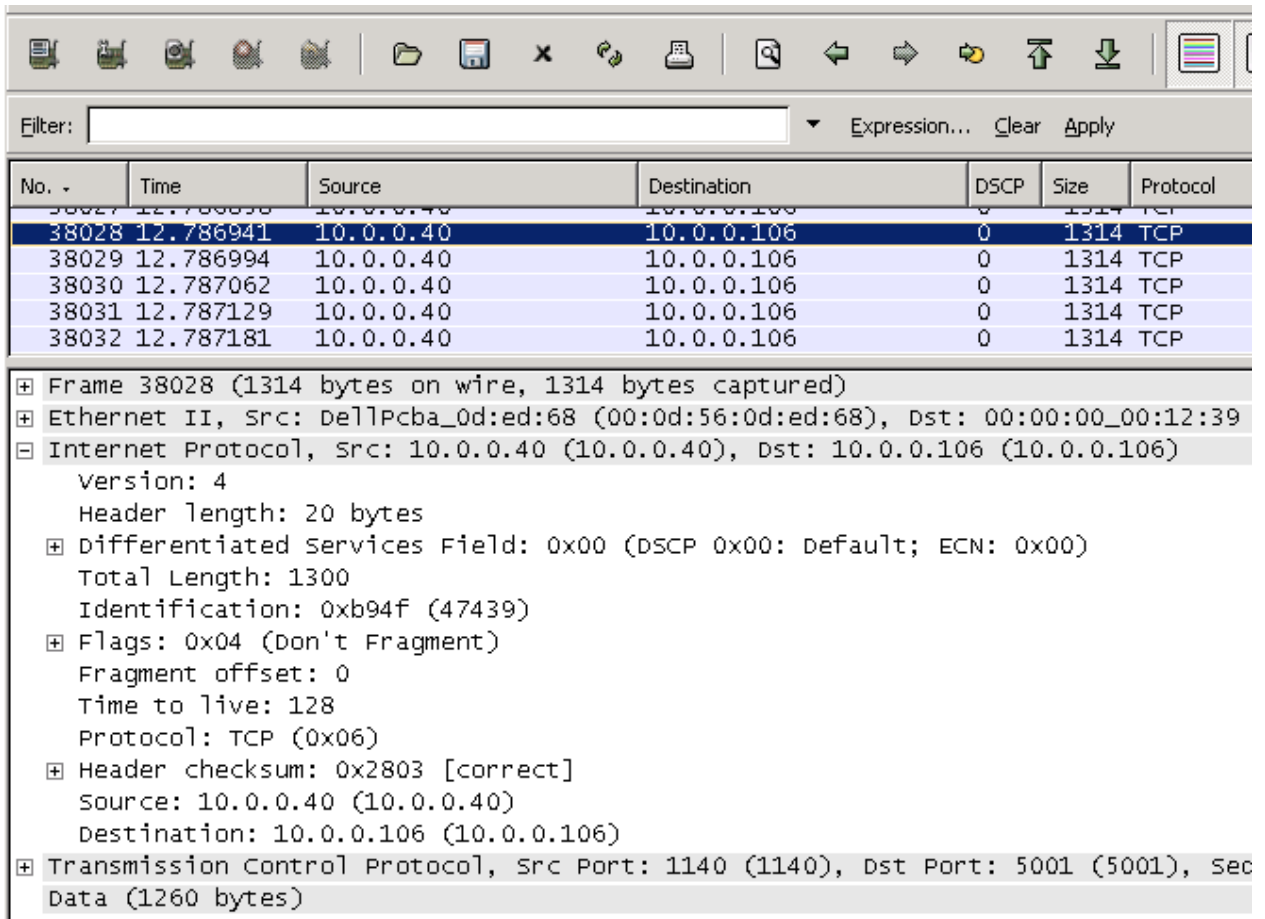
Now to run the test in the opposite direction simply reverse the commands on the client and server. To stop Iperf on the server or client side enter `Ctrl+C`.

A key point to remember when testing bandwidth with Iperf is that Iperf consumes all bandwidth available between client/server via TCP, regardless of LAN, WAN, or VPN connection. Thus, if traversing a WAN link you can quickly saturate the link, potentially effecting user/application experience! GE recommends caution in planning your testing accordingly.

Another example is a test performed over an internet VPN connection. As you might imagine, the bandwidth performance is slower which is due to factors such as encryption and Internet speed.

```
Administrator: C:\Windows\system32\cmd.exe
C:\Windows\System32>iperf -c 10.0.1.47 -i 1
-----
Client connecting to 10.0.1.47, TCP port 5001
TCP window size: 63.0 KByte (default)
-----
[156] local 10.0.0.121 port 51128 connected with 10.0.1.47 port 5001
[ ID] Interval      Transfer    Bandwidth
[156] 0.0- 1.0 sec   128 KBytes  1.05 Mbits/sec
[156] 1.0- 2.0 sec   104 KBytes  852 Kbits/sec
[156] 2.0- 3.0 sec   96.0 KBytes 786 Kbits/sec
[156] 3.0- 4.0 sec   96.0 KBytes 786 Kbits/sec
[156] 4.0- 5.0 sec   96.0 KBytes 786 Kbits/sec
[156] 5.0- 6.0 sec   96.0 KBytes 786 Kbits/sec
[156] 6.0- 7.0 sec   96.0 KBytes 786 Kbits/sec
[156] 7.0- 8.0 sec   80.0 KBytes 655 Kbits/sec
[156] 8.0- 9.0 sec   96.0 KBytes 786 Kbits/sec
[156] 9.0-10.0 sec   96.0 KBytes 786 Kbits/sec
[156] 0.0-10.6 sec   992 KBytes 764 Kbits/sec
C:\Windows\System32>_
```

Also, you can use a protocol analyzer such as Wireshark on the client/server to verify various aspects of the test, such as L4 protocol, Src/Dst ports, packet size, etc.



As shown, Iperf is a simple to use tool that can quickly provide bandwidth performance metrics over a given link. Instead of guessing or suspecting network performance is the issue, Iperf lets you know.

The examples above used basic options, but keep in mind you can change or add numerous options to suit your testing. Below is the output of the help command to show you the various things you can change.

```
Administrator: C:\Windows\system32\cmd.exe
C:\Windows\System32>iperf --help
Usage: iperf [-s|-c host] [options]
iperf [-h|--help] [-v|--version]

Client/Server:
-f, --format [kmKM] format to report: Kbits, Mbits, KBytes, MBytes
-i, --interval # seconds between periodic bandwidth reports
-l, --len #[KM] length of buffer to read or write (default 8 KB)
-m, --print_mss print TCP maximum segment size (MTU - TCP/IP header)
-o, --output <filename> output the report or error message to this specified file
-p, --port # server port to listen on/connect to
-u, --udp use UDP rather than TCP
-w, --window #[KM] TCP window size (socket buffer size)
-B, --bind <host> bind to <host>, an interface or multicast address
-C, --compatibility for use with older versions does not sent extra msgsg
-M, --mss # set TCP maximum segment size (MTU - 40 bytes)
-N, --nodelay set TCP no delay, disabling Nagle's Algorithm
-U, --IPv6Version Set the domain to IPv6

Server specific:
-s, --server run in server mode
-D, --daemon run the server as a daemon
-R, --remove remove service in win32

Client specific:
-b, --bandwidth #[KM] for UDP, bandwidth to send at in bits/sec
(default 1 Mbit/sec, implies -u)
-c, --client <host> run in client mode, connecting to <host>
-d, --dualtest Do a bidirectional test simultaneously
-n, --num #[KM] number of bytes to transmit (instead of -t)
-r, --tradeoff Do a bidirectional test individually
-t, --time # time in seconds to transmit for (default 10 secs)
-F, --fileinput <name> input the data to be transmitted from a file
-I, --stdin input the data to be transmitted from stdin
-L, --listenport # port to receive bidirectional tests back on
-P, --parallel # number of parallel client threads to run
-T, --ttl # time-to-live, for multicast (default 1)

Miscellaneous:
-h, --help print this message and quit
-v, --version print version information and quit

[KM] Indicates options that support a K or M suffix for kilo- or mega-

The TCP window size option can be set by the environment variable
TCP_WINDOW_SIZE. Most other options can be set by an environment variable
IPERF_<long option name>, such as IPERF_BANDWIDTH.
```

GE Technical Services

Please reach out to GE Technical Services if you have any issues with your configurations.
~ENERGY Digital Energy MDS Technical Support <GEMDS.techsupport@ge.com>

OR

Call +1 585.241.5510 (Technical Services Help Line)

Content Sources

1. GE MDS Tech Services custom content
2. GE MDS Product Management Documentation
3. IPerf testing - <https://www.sd-wan-experts.com/blog/iperf-bandwidth-testing/>
4. Documentation created by Tech Services

End of application bulletin.